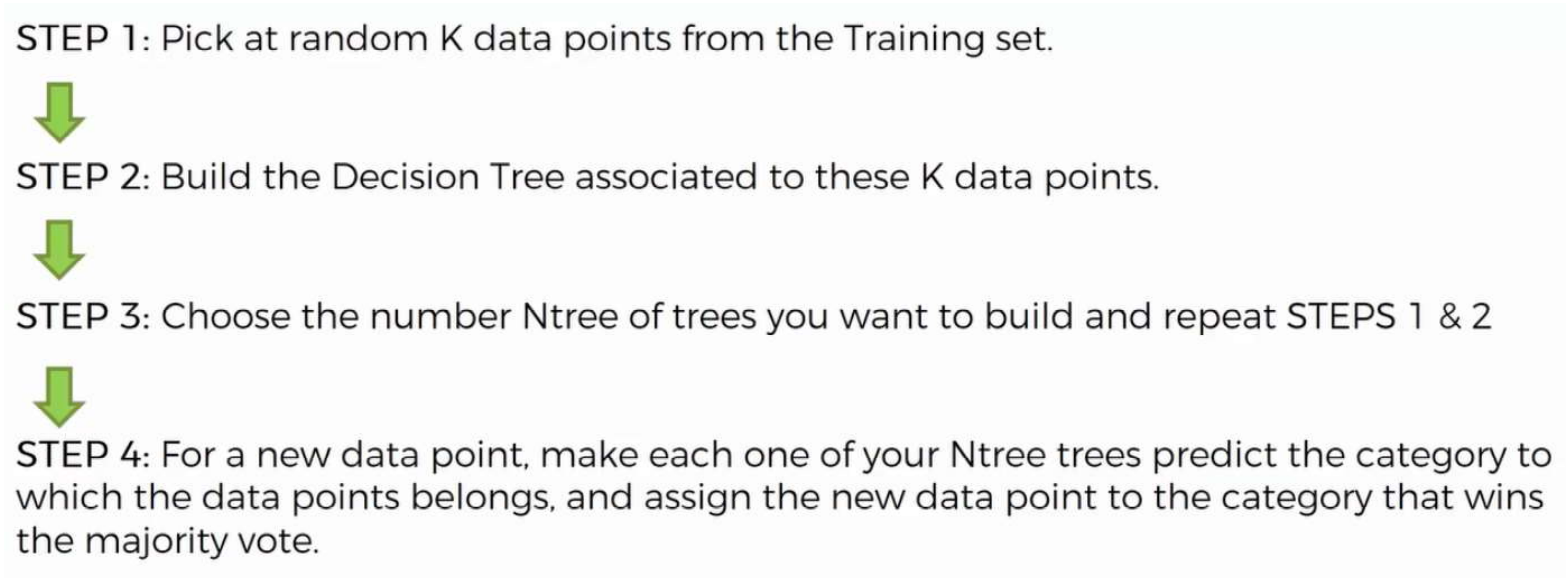
Microsoft’s Kinect

Real life Decision Tree Application Example: Microsoft's Kinect

- The device which is connected to Xbox used an infra-red grid to recognise where different body parts are located and their movements in the given space with the help of machine learning algorithm. Microsoft decided to go with the RF algorithm over other ML algorithms (available at that point in time) to develop this piece of hardware & software. Result: Faster speed, processing and reduce hardware cost
- **Source:** https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/BodyPartRecognition.pdf

Random Forest Intuition

Ensemble Learning: When you take multiple machine learning algorithm and put them together to make a bigger machine learning algorithm - the final one is leveraging many algorithm (e.g. running Decision Tree multiple times rather than once)



STEP 1: Pick at random K data points from the Training set.

STEP 2: Build the Decision Tree associated to these K data points.

STEP 3: Choose the number Ntree of trees you want to build and repeat STEPS 1 & 2

STEP 4: For a new data point, make each one of your Ntree trees predict the category to which the data points belongs, and assign the new data point to the category that wins the majority vote.

Ensemble Modeling

- Ensemble Learning
- Basic Ensemble Techniques
- Advanced Ensemble Techniques
- Algorithms

Ensemble Learning

You have bought a pair of shoes and you would like to have feedback on it. What would you do?

- You may ask one of your friends about his or her opinion
- You may ask few of your colleagues about their opinion
- You may ask few people about their opinion – some of which may be your friends, colleagues and neighbours

Ensemble Learning

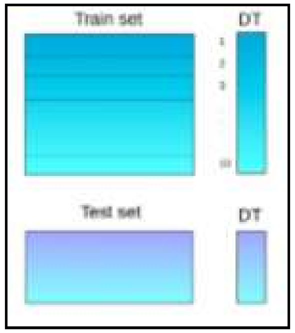
- You may ask one of your friends about his or her opinion
  - it’s very likely that the person you have chosen likes you very much and doesn’t want to break your heart by providing a 1-star rating to the horrible work you have created
- You may ask few of your colleagues about their opinion
  - This should provide a better idea about your choice but they may not be the SME as they may not have the right skills
- You may ask few people about their opinion – some of which may be your friends, colleagues and neighbours
  - This would be a more generalized and diverse opinion since now you have people with different sets of skills and it turns out to be a better approach to get the right opinion compared to the previous cases

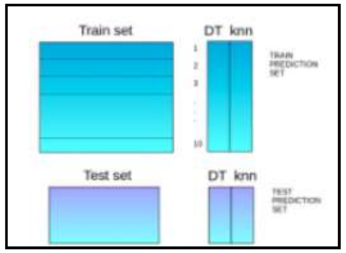
Simple Ensemble Techniques

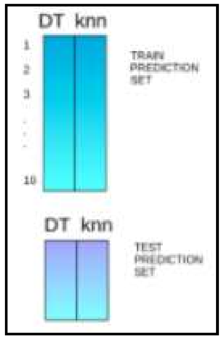
- Max Voting
  - The max voting method is generally used for classification problems. In this technique, multiple models are used to make predictions for each data point. The predictions by each model are considered as a ‘vote’. The predictions which we get from the majority of the models are used as the final prediction.
- Averaging
  - Similar to the max voting technique, multiple predictions are made for each data point in averaging. In this method, we take an average of predictions from all the models and use it to make the final prediction. Averaging can be used for making predictions in regression problems or while calculating probabilities for classification problems.
- Weighted Averaging
  - This is an extension of the averaging method. All models are assigned different weights defining the importance of each model for prediction. For instance, if two of your colleagues are critics, while others have no prior experience in this field, then the answers by these two friends are given more importance as compared to the other people.

Advanced Ensemble Techniques

- **Stacking**
  - Uses predictions from multiple models (for example Decision Tree, KNN or SVM) to build a new model. This model is used for making predictions on the test set.
  - Steps:
    1. The train set is split into 10 parts (say)
    2. A base model (e.g. Decision Tree) is fitted on 9 parts and predictions are done for the 10th part (this is done for each part of the train set)
    3. The base model is then fitted on the whole train dataset
    4. Using this model. Predictions are made on the test set
    5. Steps 2 to 4 are repeated for another base model (say KNN) resulting in another set of predictions for the train and test set
    6. Predictions from the train set are used as features to build a new model
    7. This model is used to make final predictions on the test set

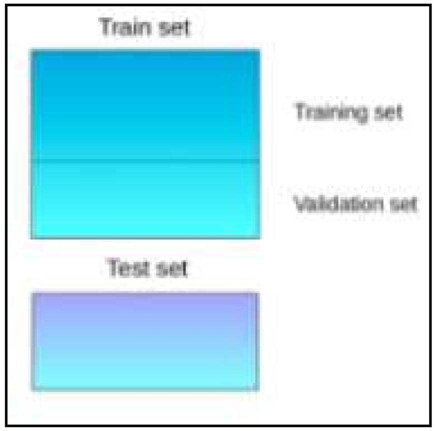


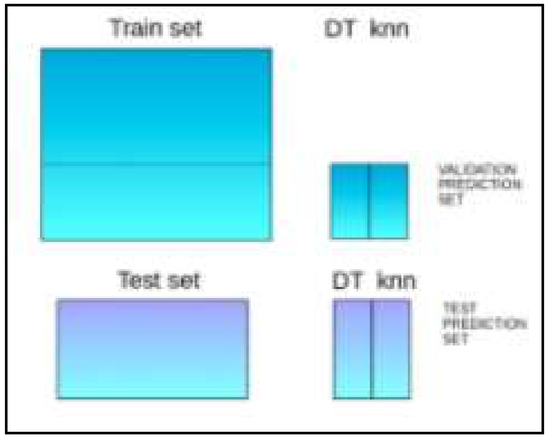




Advanced Ensemble Techniques

- **Blending**
  - Blending follows the same approach as stacking but uses only a holdout (validation) set from the train set to make predictions i.e. unlike stacking, the predictions are made on the holdout set only. The holdout set and the predictions are used to build a model which is run on the test set.
  - Steps
    1. The train set is split into training and validation sets.
    2. Model(s) are fitted on the training set.
    3. The predictions are made on the validation set and the test set.
    4. The validation set and its predictions are used as features to build a new model.
    5. This model is used to make final predictions on the test and features.



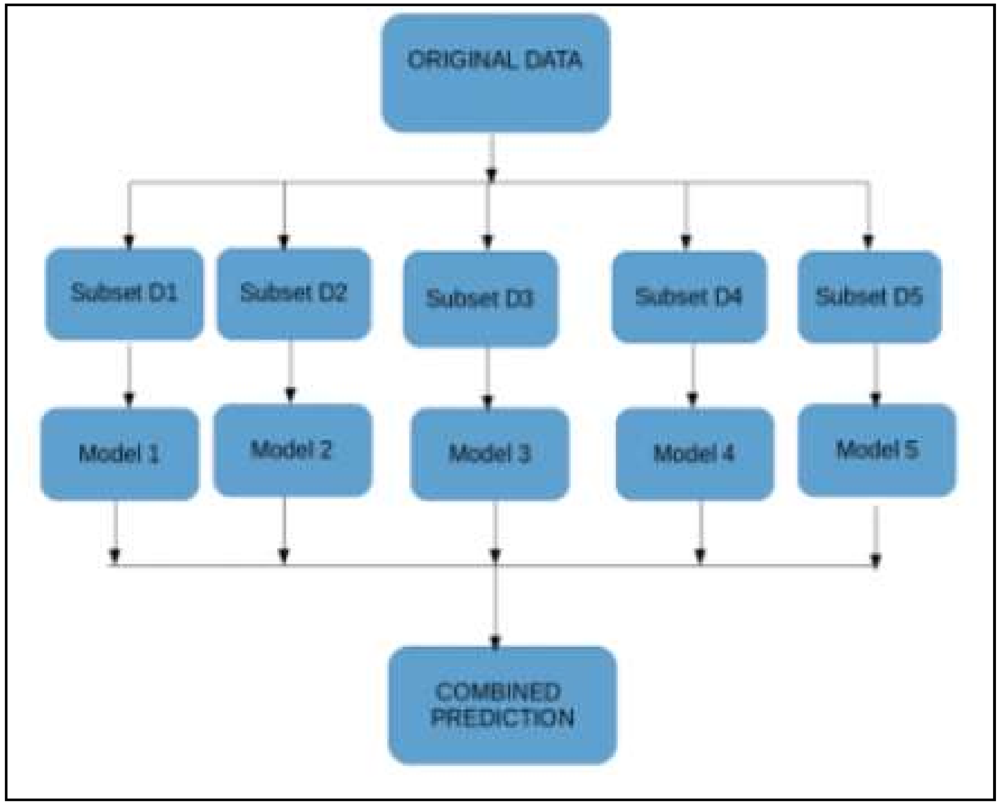


Advanced Ensemble Techniques

- **Bagging**
  - The idea behind bagging is combining the results of multiple models (for instance, all decision trees) to get a generalized result. But creating all the models on the same set of data in most likelihood give the same result as the input is the same for all. To overcome this a sampling technique in which subsets of observations from the original dataset are created with replacement [Note: the size of the subsets is the same of the original set]
  - Bagging (or Bootstrap Aggregating) technique uses these subsets (bags) to get a fair idea of the distribution (complete set).

Advanced Ensemble Techniques

- Steps
  1. Multiple subsets are created from the original dataset, selecting observations with replacement.
  2. A base model (weak model) is created on each of these subsets.
  3. The models run in parallel and are independent of each other.
  4. The final predictions are determined by combining the predictions from all the models.

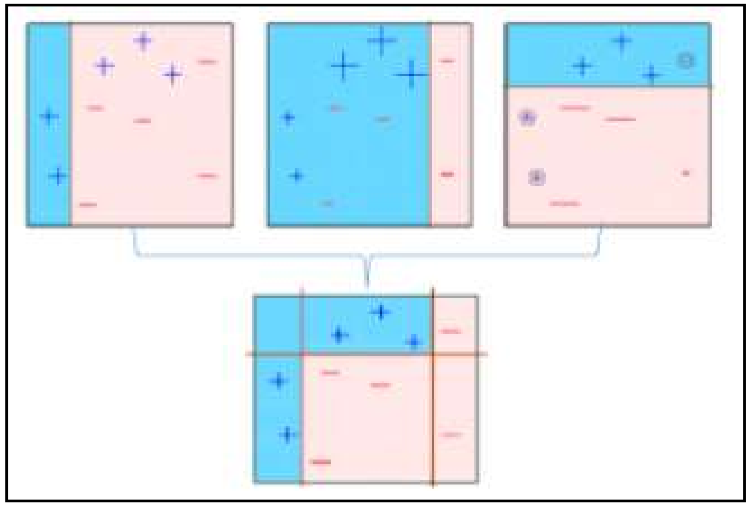


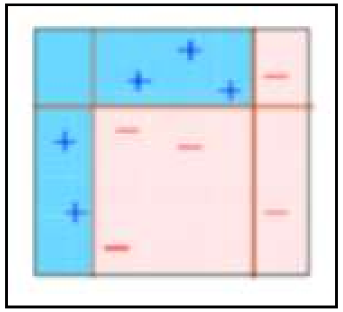
A Simple Question

- If a data point is incorrectly predicted by the first model, and then the next (probably all models), will combining the predictions provide better results?

Advanced Ensemble Techniques

- **Boosting**
  - A sequential process, where each subsequent model attempts to correct the errors of the previous model. The succeeding models are dependent on the previous model.
  - Steps
    1. A subset is created from the original dataset.
    2. Initially, all data points are given equal weights.
    3. A base model is created on this subset.
    4. This model is used to make predictions on the whole dataset.
    5. Errors are calculated using the actual values and predicted values.
    6. The observations which are incorrectly predicted, are given higher weights.
    7. Another model is created and predictions are made on the dataset.
    8. (This model tries to correct the errors from the previous model)
    9. Similarly, multiple models are created, each correcting the errors of the previous model.





Algorithms

- Bagging and Boosting are two of the most commonly used techniques in machine learning.
  - Bagging algorithms:
    - Bagging meta-estimator
    - Random forest
  - Boosting algorithms:
    - AdaBoost
    - GBM
    - XGBM
    - Light GBM
    - CatBoost

Algorithms

- **Bagging meta-estimator**
  - An ensembling algorithm that can be used for both classification (BaggingClassifier) and regression (BaggingRegressor) problems.
  - It follows the typical bagging technique to make predictions.
- Steps
  1. Random subsets are created from the original dataset (Bootstrapping).
  2. The subset of the dataset includes all features.
  3. A user-specified base estimator is fitted on each of these smaller sets.
  4. Predictions from each model are combined to get the final result.

Algorithms

- **Random Forest**
  - It is another ensemble machine learning algorithm that follows the bagging technique.
  - An extension of the bagging estimator algorithm.
  - The base estimators in random forest are decision trees.
  - Unlike bagging meta estimator, random forest randomly selects a set of features which are used to decide the best split at each node of the decision tree.
- Steps
  1. Random subsets are created from the original dataset (bootstrapping).
  2. At each node in the decision tree, only a random set of features are considered to decide the best split.
  3. A decision tree model is fitted on each of the subsets.
  4. The final prediction is calculated by averaging the predictions or max vote from all decision trees.

Parameters of Random Forest

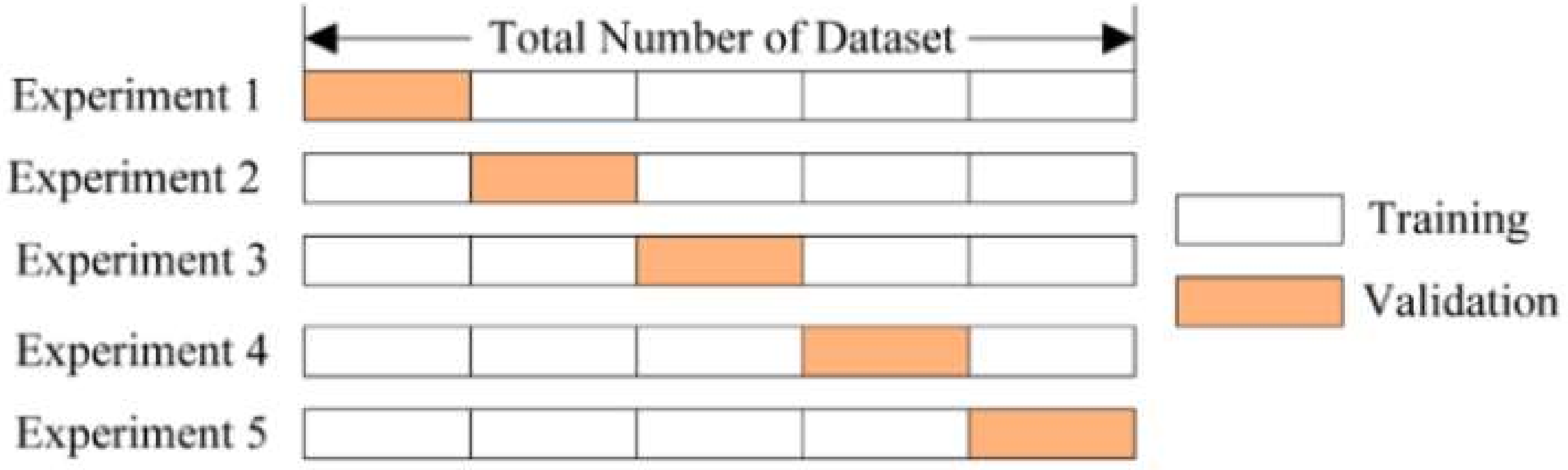
- **n_estimators:**
  - It defines the number of decision trees to be created in a random forest.
  - Generally, a higher number makes the predictions stronger and more stable, but a very large number can result in higher training time.
- **criterion**:
  - It defines the function that is to be used for splitting.
  - The function measures the quality of a split for each feature and chooses the best split.
- **max_features: [sqrt(# of Features)]**
  - It defines the maximum number of features allowed for the split in each decision tree.
  - Increasing max features usually improve performance but a very high number can decrease the diversity of each tree.
- **max_depth**:
  - Random forest has multiple decision trees. This parameter defines the maximum depth of the trees.
- **min_samples_split: [8-12%]**
  - Used to define the minimum number of samples required in a leaf node before a split is attempted.
  - If the number of samples is less than the required number, the node is not split.
- **min_samples_leaf: [2-4%]**
  - This defines the minimum number of samples required to be at a leaf node.
  - Smaller leaf size makes the model more prone to capturing noise in train data.
- **max_leaf_nodes:**
  - This parameter specifies the maximum number of leaf nodes for each tree.
  - The tree stops splitting when the number of leaf nodes becomes equal to the max leaf node.
- **random_state**:
  - This parameter is used to define the random selection.
  - It is used for comparison between various models.

Hyper-parameter Tuning

- Relies more on experimental results than theory, and thus the best method to determine the optimal settings is to try many different combinations evaluate the performance of each model.
- Evaluating each model only on the training set can lead to one of the most fundamental problems in machine learning: <u>Overfitting</u>
- Note: When a model performs highly on the training set but poorly on the test set, this is known as overfitting, or essentially creating a model that knows the training set very well but cannot be applied to new problems.
- An overfit model may look impressive on the training set, but will be useless in a real application.
- So what’s the solution?
- The solution is Cross Validation

Cross Validation

- The dataset is split into a training and a testing set.
- In K-Fold CV, we further split our training set into K number of subsets, called folds.
- Iteratively the model is fit K times, each time training the data on K-1 of the folds and evaluating on the Kth fold (called the validation data).



Example

n-estimators	[100,150,200]		3	
max_depth	[3,5,7,9]		4	
min_sample_leaf	[30,50,75,100]		4	
min_sample_split	[90,150,300,500]		4	
max_features	[5,7]		2	3*4*4*4*2
				384
CV	10		10	384*10
				3840

OOB

- While training each tree in **random forest**, all the observations are not used. So for each bag, those unused observations can be used to find the prediction **error** for that particular bag i.e. **OOB Error is the number of wrongly classifying the OOB Sample.** The **OOB error** rate can then be obtained by averaging the prediction **error** from all the bags.
- Similarly, each of the **OOB** observations is passed through every DT that did not contain the **OOB** observations in its bootstrap training data and a majority prediction is noted for each row. And lastly, the **OOB score** is computed as **the number of correctly predicted rows from the out of bag sample**. <u>It is a way of validating the Random forest model.</u>
- **Note:** As only a subset of DTs is used for determining the OOB score. This leads to reducing the overall aggregation effect in bagging. Thus in general, validation on a full ensemble of DTs is better than a subset of DT for estimating the score.

OOB

- If there are N rows in the training data set. Then, the probability of not picking a row in a random draw is: (N-1) / N
- Using sampling-with-replacement the probability of not picking N rows in random draws is; ((N-1) / N)^N
- Which in the limit of large N becomes equal to:

$$\lim_{N\to\infty}\left(1-\frac{1}{N}\right)^{N} = e^{-1} = 0.368$$

- Therefore, about 36.8 % of total training data are available as OOB sample for each DT and hence it can be used for evaluating or validating the random forest model.

- **Advantages of using OOB_Score:**
  - **<u>No leakage of data</u>:** Since the model is validated on the OOB Sample, which means data hasn't been used while training the model in any way, so there isn't any leakage of data and henceforth ensures a better predictive model.
  - **<u>Less Variance</u> : [**More Variance ~ Overfitting due to more training score and less testing score]. Since OOB_Score ensures no leakage, so there is no over-fitting of the data and hence least variance.
  - **<u>Better Predictive Model</u>:** OOB_Score helps in the least variance and hence it makes a much better predictive model than a model using other validation techniques.
  - **<u>Less Computation</u>:** It requires less computation as it allows one to test the data as it is being trained.

- **Disadvantages of using OOB_Error :**
  - **<u>Time Consuming</u>:** The method allows to test the data as it is being trained, but the overall process is a bit time-consuming as compared to other validation techniques.
  - **<u>Not good for Large Datasets</u>:** As the process can be a bit time-consuming in comparison with the other techniques, so if the data size is huge, it may take a lot more time while training the model.
  - **<u>Best for Small and medium-size datasets</u>:** Even if the process is time-consuming, but if the dataset is medium or small sized, OOB_Score should be preferred over other techniques for a much better predictive model.

- Random Forest can be a very powerful technique for predicting better values if we use the OOB_Score technique. Even if OOB_Score takes a bit more time but the predictions are worth the time consumed in training the random forest model with the OOB_Score parameter set as True.

Summary: Random Forest

- Random forest is a kind of universal machine learning technique
- It can be used for both regression (target is a continuous variable) or classification (target is a categorical variable) problems
- It also works with columns of any kinds, like pixel values, zip codes, revenue, etc.
- In general, random forest does not overfit (it's very easy to stop it from overfitting)
- You do not need a separate validation set in general. It can tell you how well it generalizes even if you only have one dataset
- It has few (if any) statistical assumptions (it doesn't assume that data is normally distributed, data is linear, or that you need to specify the interactions)
- Requires very few feature engineering tactics, so it's a great place to start. For many different types of situations, you do not have to take the log of the data or multiply interactions together

- Most machine learning models (including random forest) cannot directly use categorical columns. We need to convert these columns into numbers first.

- *RandomForestRegressor* and *RandomForestClassifier* functions are used in Python for regression and classification problems respectively.